3调度算法

2018年10月22日 9:00

-
- 作业和作业调度

1. 多道批处理系统中，作业是用户提交给系统的一项相对独立的工作

一. 批处理系统中的作业

1. 作业和作业步
   1) 作业(Job)除了包含通常的程序和数据， 还应配有一份作业说明书，系统根据说明书控制程序的运行
      (1) 批处理系统中，以作业为基本单位从外存调入内存
   2) 作业步(Job Step)：作业的每一个加工步骤
      (1) 往往把一个作业步的输出作为下一个作业步的输入。 例如，一个典型的作业可分成三个作业步：编译、连接装配、运行
2. 作业控制块(Job Control Block)
   1) 是作业在系统中存在的标志
   2) 包含：标识符、用户名、用户帐户、作业类型(CPU 繁忙型、I/O 繁忙型、批量型、终端型)、作业状态、调度信息(优先级、作业已运行时间)、资源需求(预计运行时间、 要求内存大小、要求 I/O 设备类型和数量等)、进入系统时间、开始处理时间、作业完成时间、作业退出时间、资源使用情况等
   3) 由系统的“作业注册”程序建立JCB，排到后备队列；由系统的调度程序按算法调度它们进入内存；执行时系统根据JCB和说明书对作业进行控制；结束后由系统回收资源，撤销JCB
3. 作业运行的三个阶段和三种状态
   1) 收容阶段：作业输入到硬盘，创建JCB，放入后备队列
      (1) 此时是后备状态
   2) 运行阶段：分配资源，建立进程，放入就绪队列
      (1) 第一次就绪到运行结束前都是运行状态
   3) 完成阶段：系统的“终止作业”程序回收JCB和资源，并将运行结果信息形成文件并输出
      (1) 运行结束后进入完成状态

二. 作业调度的主要任务

1) 作业调度的主要功能是根据JCB，审查系统资源，从外存的后备队列中选取某些作业调入内存，并创建进程、分配资源，再插入就绪队列。有时把作业调度称为接纳调度(Admission Scheduling)

1. 接纳多少作业，取决于多道程序度(Degree of Multiprogramming)
   1) 多道程序度：允许多少个作业同时在内存中运行
   2) 太多容易内存不足而中断运行；太少使平均周转时间显著延长
   3) 取决于系统规模、运行速度、作业大小、系统性能
2. 接纳哪些作业，取决于采用的调度算法
   1) 最易：FCFS；最常用：SJF；较常用：PSA；最好：HRRN
   2) 作业进入批处理系统后总是先驻留在外存的后备队列上，因而需要作业调度。分时系统无需作业调度，直接将命令或数据送入内存以实现及时响应，只需用接纳控制措施限制进入系统的用户数

三. FCFS和SJF

1. **先来先服务**调度算法(First-come first-served)
   1) 相当于只考虑等待时间最长的作业
   2) 可组合使用，如先按优先级设多个队列，每个队列里FCFS

2. **短作业优先**(Short job first)
    1) 长短以作业要求的运行时间来衡量，多数是短的
    2) 缺点：
        (1) 运行时间估计短了会提前终止作业
        (2) 不考虑等待时间，长作业周转时间明显增长，出现饥饿现象
        (3) 无法实现人机交互
        (4) 不考虑紧迫度，高紧迫性作业不能保证及时处理

四. PSA和HRRN

1. 优先级调度算法(Priority-scheduling algorithm)，详见进程调度
    1) 外部基于作业紧迫度，赋予作业一个优先级，保证高紧迫性作业先运行
2. **高响应比优先调度**算法(High Response Ratio Next)
    1) 是唯一一个基本只用于作业调度的算法
    2) $$\text{优先权} = \frac{\text{等待时间} + \text{要求服务时间}}{\text{要求服务时间}}$$
    3) $$R_P = \frac{\text{等待时间} + \text{要求服务时间}}{\text{要求服务时间}} = \frac{\text{响应时间}}{\text{要求服务时间}}$$
    4) 以上两个公式本质上是一个，但前者称为优先权，后者称为**响应比**
    5) 实现了FCFS和SJF的折中：等待时间差不多时，类似SJF；要求服务时间差不多时类似FCFS；长作业不会饥饿
    6) 同时考虑了等待时间和运行时间，改善了处理机调度性能，但每次调度前都要计算Rp增加了系统开销
        -
        - 进程调度

一. 进程调度的任务、机制和方式

1. 进程调度的任务：保存处理机现场、按算法选取进程、把处理器分配进程
2. 进程调度机制
    1) 排队器：把转为就绪状态的进程插入对应就绪队列
    2) 分派器：从就绪队列取出应调度的进程，并选新进程的上下文切换
    3) 上下文切换器：
        (1) 保存当前进程的上下文到PCB，再装入分派程序的上下文
        (2) 移出分派程序的上下文，把新进程的CPU现场装入各CPU寄存器
        (3) 需要执行大量load和store等操作指令，执行一次上下文切换大约可执行上千条其他指令
        (4) 或用两套硬件分别供系统态、应用程序使用，则上下文切换只需改变指针指向不同寄存器组
3. 进程调度方式
    1) 非抢占方式(Nonpreemptive Mode)
        (1) 决不会因为时钟中断等原因而抢占正在运行进程的处理机
        (2) 引起调度的原因：执行完毕，或因发生某事件不能继续执行；因提出 I/O 请求而暂停执行；执行了 wait、Block 、Wakeup 等原语

2) 抢占方式(Preemptive Mode)
   (1) 允许调度程序根据以下原则去暂停正在执行的进程
      i. 优先权原则、短作业(进程)优先原则、时间片原则
   (2) 实现了分时系统的人机交互、实时系统的HRT

二. **轮转调度**算法Round Robin

1) 分时系统中基于时间片的轮转使n个就绪进程都约获得1/n的处理机时间

1. 轮转法的基本原理
   1) 按FCFS排就绪队列，隔一定时间(如30ms)产生一次中断，调度队首进程
   2) 老师说默认刚结束的进程出现在队尾，即新到进程可能在倒数第二
2. 进程切换时机：执行中程序提前完成、时间片结束（然后移至队尾）
3. 时间片大小的确定
   1) 太短导致频繁调度、切换，增加系统开销
   2) 太长相当于退化成FCFS，无法满足短作业和交互
   3) 因此时间片一般取略大于一次交互的时间，缩小响应时间

三. **优先级调度**算法Priority-scheduling algorithm

1) 轮转法没有考虑紧迫性，即默认紧迫性相同

1. 调度算法类型
   1) 非抢占式：直至执行完成或自行放弃，都不重新分配处理机
   2) 抢占式：一旦出现新进程，立刻比较优先级，若高于执行中进程，立即重新分配处理器。适用于实时性要求高的系统
2. 优先级的类型
   1) 静态优先级：创建进程时确定，是一个不变的整数
      (1) 确定依据：
         i. 进程类型：如系统进程优先权高(接收、对换、磁盘 I/O 等进程)
         ii. 进程对资源的要求：要求少的，优先级高
         iii. 用户要求：紧迫度及用户所付费用的多少
      (2) 简单易行、系统开销小、不精确、低优先级进程可能饥饿
   2) 动态优先级：先赋初值，再随进程推进/时间增加而改变
      (1) 相同初值时，类似FCFS
      (2) 若随等待时间增加而增加优先级，则短优先级能获得处理机
      (3) 若采用抢占式，还能防止长作业长期垄断处理机

四. 多队列调度算法

1. 将不同类型或性质的进程分配在不同的就绪队列，不同队列采取不同调度算法，不同进程可以设置不同优先级，不同队列也可以有不同优先级
2. 多处理机系统可以为每个处理机设置一个单独的就绪队列
3. 需相互合作的进程或线程也可分配到一组处理机的多个就绪队列，实现并行

五. **多级反馈队列**(multileved feedback queue)调度算法

1) 它不用事先安排各进程所需的执行时间，公认较好的进程调度算法

1. 调度机制
   1) 设置多个就绪队列，优先级逐个降低，时间片逐个增倍

(1)

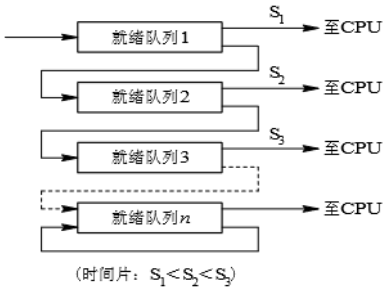


图 3-7 多级反馈队列调度算法

2) 各队列内采用FCFS，新进程放入第一个队列末尾。时间片结束后若未完成，则进入下一队列末尾，到某个队列N后开始轮转

3) 按队列优先级调度，即只有1~(i-1)队列都为空时才调用i队列首进程。若有新进程进入高优先级队列，应立即把当前进程放入原队列末尾

2. 调度算法的性能

1) 一般规定第1队列时间片略大于交互所需处理时间，满足用户需求

(1) 终端型用户：交互型小作业，一般第1队列就能完成

(2) 短批处理作业用户：一般在前三队列都能完成，周转时间仍短

(3) 长批处理作业用户：第n队列中RR，一般不必担心长期无法处理

六. **基于公平原则**的调度算法

1) 其他算法并不保证作业占用多少处理机时间，不一定公平

1. 保证调度算法：向用户保证性能，n个同类型进程各获得处理机时间1/n，需要这些功能：

1) 跟踪计算每个进程实际获得的处理时间

2) 计算应获得的处理时间，即创建以来的时间除以进程数n

3) 计算执行时间比率，即1）/2）

4) 比较各进程的比率

5) 将处理机分配给比率最小的进程，运行到超过第二小的比率

2. 公平分享调度算法

1) 若各用户所拥有的进程数不同，保证调度算法对用户就不公平

2) 同一用户有多个进程，则该用户的每个进程活动的时间会减少

-
- 实时调度

一. 实现实时调度的基本条件

1. 提供必要信息

1) 就绪时间：成为就绪态的起始时间、周期任务是预知的一串时间序列

2) 开始截止时间和完成截止时间，一般只需一个

3) 处理时间：开始执行到完成所需时间

4) 资源要求：执行时所需的一组资源

5) 优先级：错过截止时间会引起故障，则有“绝对”优先级，无重大影响的就赋予“相对”优先级，供调度程序参考。（即硬/软实时任务）

2. 系统处理能力强

1) m 个周期性的硬实时任务，处理时间为 Ci，周期时间为 Pi， 在单处理

机情况下，必须满足下面的限制条件，系统才可调度

(1) $$\sum_{i=1}^{m}\frac{C_i}{P_i}\leqslant 1$$

2) 若是N个处理机的多处理机系统，限制条件改为

(1) $$\sum_{i=1}^{m}\frac{C_i}{P_i}\leqslant N$$

3) 都只是最低要求，需要留有余地给调度算法、任务切换、消息传递等

3. 采用抢占式调度机制：满足HRT任务
   1) 如果已知任务开始截止时间，也可用非抢占机制，执行完关键程序和临界区后，进程及时阻塞自己
4. 具有快速切换机制
   1) 对外部中断的快速响应能力：快速硬件中段机构，保证紧迫任务
   2) 快速的任务分派能力：减少切换任务的时间开销

二. 实时调度算法的分类

1. 非抢占式调度算法
   1) 非抢占轮转：时间片结束后挂在轮转队列末
      (1) 偏软，有数秒至数十秒的响应时间
   2) 非抢占优先：新到的高优先级排在队首
      (1) 偏硬，响应时间数秒至数百毫秒
2. 抢占式调度算法
   1) 基于时钟中断的抢占式优先级：等时钟中断发生时调度高优先级任务
      (1) 偏软，调度延迟几十至几毫秒
   2) 立即抢占优先级(Immediate Preemption)：快速响应外部事件中段，只要当前任务未处于临界区，就能立即剥夺处理器给请求中断的紧迫任务

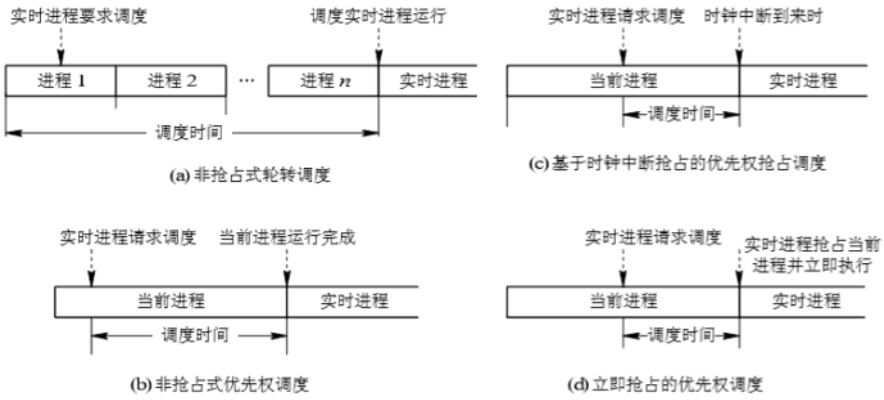


图 3-8 实时进程调度

三. **最早截止时间优先**(Earliest Deadline First)算法

1) 截止时间越早，优先级越高

1. 非抢占：用于非周期实时任务（假设截止时间3<4<2）

1)

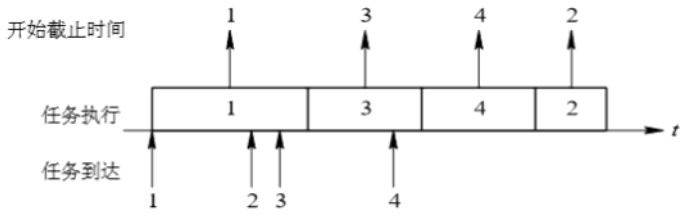


图 3-9 EDF 算法用于非抢占调度的调度方式

2. 抢占：用于周期实时任务（第一行是要求，二三行是任务固定优先级，第四行是正解）

1)

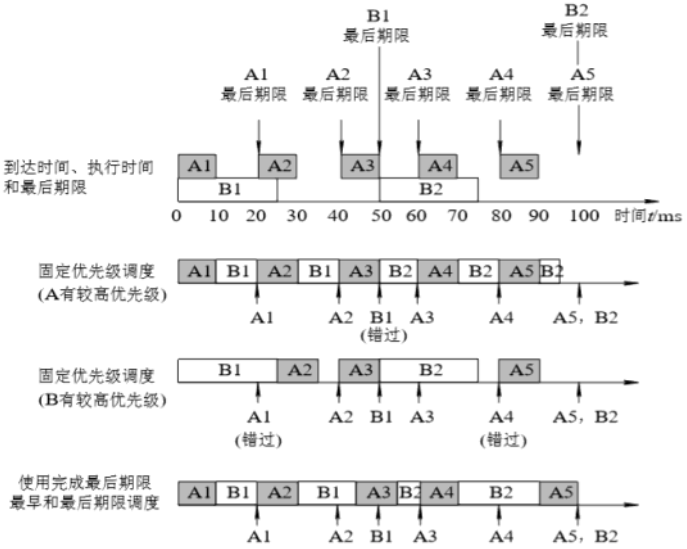


图 3-10 最早截止时间优先算法用于抢占调度方式之例

四. **最低松弛度优先**(Least Laxity First)算法

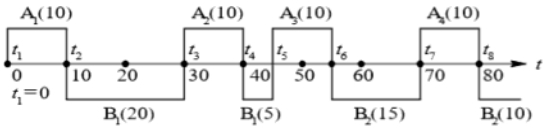
1. 紧急程度=**松弛度=必须完成时间-其本身的运行时间-当前时间**
   1) 相当于距离deadline还剩的时间
2. 主要用于可抢占调度，完成周期性实时任务
3. 老师们认为每当有新进程进入，就应当比较松弛度，但图上是每当有进程松弛度为零才调度
4. 

图 3-12 利用 LLF 算法进行调度的情况

五. 优先级倒置问题(priority inversion problem)

1. 优先级倒置的形成：低优先级进程/线程，延迟或阻塞了高优先级进程/线程。如占用了互斥资源
2. 其解决方法
   1) 进入临界区后不允许处理机被抢占。仍可能导致高优先级进程等待很久
   2) 动态优先级继承：若低优先级进程抢占了高优先级进程想要的资源，则让低优先级进程继承到该阻塞进程的优先级，防止被中优先级插入

i.

ii.

iii.

iv.

v.

vi.

vii. --------我是底线----------